Simple Complexity Analysis of Direct Search

Simple Complexity Analysis of Direct Search

Jakub Konečný * Peter Richtárik †

School of Mathematics
University of Edinburgh
United Kingdom

September 30, 2014

Abstract

We consider the problem of unconstrained minimization of a smooth function in the derivative-free setting. In particular, we study the pattern search method (of directional type). Despite relevant research activity spanning several decades, until recently no complexity guarantees—bounds on the number of function evaluations needed to find a satisfying point—for methods of this type were established. Moreover, existing complexity results require long proofs and the resulting bounds have a complicated form. In this paper we give a very brief and insightful analysis of pattern search for nonconvex, convex and strongly convex objective function, based on the observation that what is in the literature called an "unsuccessful step", is in fact a step that can drive the analysis. We match the existing results in their dependence on the problem dimension (n) and error tolerance (ϵ), but the overall complexity bounds are much simpler, easier to interpret, and have better dependence on other problem parameters. In particular, we show that the number of function evaluations needed to find an ϵ-solution is $O(n^2/\epsilon)$ (resp. $O(n^2\log(1/\epsilon))$) for the problem of minimizing a convex (resp. strongly convex) smooth function. In the nonconvex smooth case, the bound is $O(n^2/\epsilon^2)$, with the goal being the reduction of the norm of the gradient below ϵ.

1 Introduction

In this work we study the problem of unconstrained minimization of a smooth function $f:\mathbb{R}^n\to\mathbb{R}$:

$$\min_{x\in\mathbb{R}^n} f(x). \tag{1}$$

Naturally, we assume that f is bounded below, and let

$$f^* \stackrel{\text{def}}{=} \inf_{x\in\mathbb{R}^n} f(x). \tag{2}$$

In particular, we consider nonconvex, convex and strongly convex objective functions f and assume that we only have access to a function evaluation oracle. That is, we work in the derivative-free setting.

*School of Mathematics, University of Edinburgh, United Kingdom (e-mail: j.konecny@sms.ed.ac.uk)

†School of Mathematics, University of Edinburgh, United Kingdom (e-mail: peter.richtarik@ed.ac.uk)

The work of both authors was supported by the Centre for Numerical Algorithms and Intelligent Software (funded by EPSRC grant EP/G036136/1 and the Scottish Funding Council). The work of J.K. was also supported by a Google European Doctoral Fellowship.

Direct search. The focus of this work is *not* on obtaining improved complexity results for derivative-free optimization as such. Instead, our goal is more moderate. We study a novel variant of an old method—*direct search*. Despite the effort by a community of researchers spanning more than half a century [12, 20, 18, 9, 4, 13, 1, 2, 3], complexity bounds for direct search have not been established until very recently [8, 11, 19]. On the other hand, existing complexity results require long proofs, with the bounds being complicated and hard to interpret, and depending on a large number of parameters of the method.

Contributions. We provide a *surprisingly brief* and unified analysis of the method when applied to a nonconvex, convex or a strongly convex function f. Previously, separate papers were required to deal with each such case. As a byproduct of the simplicity of our analysis we obtain compact and easy to interpret complexity bounds, with small constants, improving on existing bounds for the direct search method. Moreover, existing bounds hold only after a specific event during the running of the algorithm is observed, while in our work we provide bounds that hold from the start, as one would expect.

In contrast with standard direct search methods, our method depends on a single parameter: a forcing constant $c > 0$. As presented in Section 2, our method seems to depend on an additional parameter: stepsize $\alpha_0 > 0$. However, we show in Section 5 that one can, at low cost, identify suitable α_0 automatically. We show that setting this parameter to the Lipschitz constant of the gradient of f optimizes the complexity bound – in this sense, despite appearance, this is the *true* stepsize parameter.

In the table below we summarize the complexity results obtained in this paper. In all cases we assume that f is L-smooth and bounded below; the assumptions listed in the first column are *additional* to this.

Assumptions on f	Goal	# function evaluations	Theorem
no additional assumptions	$\|\nabla f(x)\| < \epsilon$	$O\left(\frac{n^2 L(f(x_0)-f^*)}{\epsilon^2}\right)$	5
convex, $\exists$ minimizer x_*, $R_0 < +\infty$	$f(x) - f(x_*) \leq \epsilon$	$O\left(\frac{n^2 L R_0^2}{\epsilon}\right)$	6
λ-strongly convex	$f(x) - f(x_*) \leq \epsilon$	$O\left(\frac{n^2 L}{\lambda} \log\left(\frac{1}{\epsilon}\right)\right)$	7

The quantity R_0 measures the size of a specific level set of f. Definitions of all quantities appearing in the table are given in Section 3.

Derivative-free optimization. It is well known [14, Section 1.2.3] that for the problem of unconstrained minimization of a smooth (and not necessarily convex) function, gradient descent takes at most $\mathcal{O}(1/\epsilon^2)$ iterations to drive the norm of the gradient below ϵ. Such a bound has been proved tight in [5]. In the context of derivative-free methods, Nesterov's random Gaussian approach [15] attains the complexity bound $\mathcal{O}(n^2/\epsilon^2)$. Vicente matches this result with a (deterministic) direct

search algorithm [19], and so does our analysis of direct search. Cartis et al. [6] derived a bound of $\mathcal{O}(n^2/\epsilon^{3/2})$ for a variant of their adaptive cubic overestimation algorithm using finite differences to approximate derivatives. In this setting, Ghadimi and Lan [**?**] achieve better (linear) dependence on n by considering a slightly more special class of problems.

In the convex case, gradient descent achieves the improved bound of $\mathcal{O}(1/\epsilon)$ [14, Section 2.1.5]. For derivative-free methods, this rate is also achievable by Nesterov's random Gaussian approach [15] and by direct search [8]. The bound on function evaluations for both methods becomes $\mathcal{O}(n^2/\epsilon)$, which we match in this paper.

If we drop the usual relaxation requirement (monotonicity of function values), Nesterov [14, Section 2.2.1] proved that the accelerated gradient descent method achieves the bound of $\mathcal{O}(1/\epsilon^{1/2})$ iterations. The derivative-free analogue of this method [15] needs $\mathcal{O}(n/\epsilon^{1/2})$ function evaluations. There are no results on direct search methods that would attain this bound.

In the strongly convex setting, gradient descent achieves linear convergence, i.e., the bound on number of iterations is $\mathcal{O}(\log(1/\epsilon))$. This rate is also achievable in derivative-free setting by multiple methods [8, 15, 7], including our version of direct search.

A recent work of Recht et al. [16] goes beyond the zero-order oracle. Central in their work is a pairwise comparison oracle, that returns only the order of function values at two different points. They provide lower and upper complexity bounds for both deterministic and stochastic oracles. A related randomized coordinate descent algorithm is proposed, that also achieves $\mathcal{O}(n\log(1/\epsilon))$ calls of the oracle for strongly convex functions. Duchi et al. [10] prove tight bounds for online bandit convex optimization problems with multi-point feedback. However, the optimal iteration complexity for single point evaluation still remains an open problem. Yet another related approach, where one has access to partial derivatives, is the randomized coordinate descent method [17]. The iteration complexity of the method is $\mathcal{O}(n/\epsilon)$ in the convex case and $\mathcal{O}(n\log(1/\epsilon))$ in the strongly convex case. This method can be extended to the derivative-free setting by considering finite difference approximation of partial derivatives.

1.1 Outline

The paper is organized as follows. In Section 2 we describe the algorithm, in Section 3 we state the three complexity theorems covering the nonconvex, convex and strongly convex cases and provide a brief discussion of the results. The theorems are proved in Section 4. Finally, we describe two initialization strategies in Section 5.

2 Direct Search Method

In this section we describe our algorithm (Algorithm 1).

The method works with a fixed finite set D of nonzero vectors in $\mathbb{R}^n$: the algorithm is only allowed to take steps of positive lengths, along directions $d \in D$. That is, every update step is of the form $x \leftarrow x + \alpha d$, for $\alpha > 0$ and $d \in D$. Clearly, D needs to be rich enough so that every point in $\mathbb{R}^n$ (in particular, the optimal point) is potentially reachable by a sequence of such steps. We shall formalize this requirement in the next section - this suffices for our discussion here.

The method starts with an initial iterate $x_0 \in \mathbb{R}^n$ and an initial stepsize parameter $\alpha_0 > 0$. Given x_{k-1} and α_{k-1}, we seek to determine the next iterate x_k. This is done as follows. First, we initialize our search for x_k by setting $x_k^0 = x_{k-1}$ and decrease the stepsize parameter: $\alpha_k = \frac{\alpha_{k-1}}{2}$.

Algorithm 1 Direct Search Method (DSM)

1. INPUT: starting point $x_0 \in \mathbb{R}^n$; stepsize $\alpha_0 > 0$; positive spanning set D; forcing constant $c > 0$

2. For $k \geq 1$ repeat

   - Set $x_k^0 = x_{k-1}$ and $\alpha_k = \frac{1}{2}\alpha_{k-1}$
   - Let $x_k^0, \ldots, x_k^{l_k}$ be generated by

     $$x_k^{l+1} = x_k^l + \alpha_k d_k^l, \quad d_k^l \in D, \quad l = 0, \ldots, l_k - 1,$$

     so that the following relations hold:

     $$f(x_k^{l+1}) \leq f(x_k^l) - c\alpha_k^2, \quad l = 0, \ldots, l_k - 1, \tag{3}$$

     and

     $$f(x_k^{l_k} + \alpha_k d) > f(x_k^l) - c\alpha_k^2 \quad \text{for all} \quad d \in D. \tag{4}$$

   - Set $x_k = x_k^{l_k}$

Having done that, we try to find $d \in D$ for which the following sufficient decrease condition holds:

$$f(x_k^0 + \alpha_k d) < f(x_k^0) - c\alpha_k^2.$$

If such d exists, we call it d_k^0, declare the search step *successful* and let $x_k^1 = x_k^0 + \alpha_k d_k^0$. Note that the identification of x_k^1 requires, in the worst case, $|D|$ function evaluations (assuming $f(x_0)$ was already computed before). This process is repeated until we are no longer able to find a successful step. That is, the process is repeated until we find $x_k^{l_k}$ which satisfies (4) (i.e., no step of stepsize α_k leads to sufficient decrease – all possible steps are *unsuccessful*). Such a point must exist since we assume that f is bounded below, and hence it is not possible to keep decreasing the function value by a constant. This way, we produce the sequence

$$x_k^0, x_k^1, \ldots, x_k^{l_k},$$

with the property that it is no longer possible to take a step from $x_k^{l_k}$ along direction $d \in D$ of length α_k which would be successful (i.e., the criterion (4) is satisfied). Having done that, we then set $x_k = x_k^{l_k}$ and proceed to the next iteration.

Note that it is possible for l_k to be equal to 0, in which case we have $x_k = x_{k-1}$. However, there is still progress, as the method has learned that the stepsize α_k does not lead to a successful step.

2.1 First observations

Having computed $f(x_{k-1})$, the method needs to perform at most $|D|(l_k + 1)$ function evaluations to identify x_k. Hence, in order to produce the sequence $x_0, \ldots, x_k$, the method needs to perform at most

$$N(k) \stackrel{\text{def}}{=} 1 + \sum_{j=1}^{k} |D|(l_j + 1) \tag{5}$$

function evaluations.

The following simple result holds *without any assumptions* on f or D apart from requiring that f be bounded below.

Proposition 1. *Algorithm 1 produces a non-increasing sequence of iterates* $\{f(x_k)\}_{k\geq 0}$. *Moreover, for each* $k \geq 1$ *we have*

$$l_k \leq \frac{f(x_{k-1}) - f(x_k)}{c\alpha_k^2} \leq \frac{f(x_0) - f^*}{c\alpha_k^2}. \tag{6}$$

and the total number of function evaluations up to iteration k *can be bounded by*

$$N(k) \leq 1 + k|D| + \frac{4(4^k - 1)|D|}{3c\alpha_0^2}(f(x_0) - \inf_{x\in\mathbb{R}^n} f(x)). \tag{7}$$

Proof. For each $k \geq 1$ we have

$$f(x_k) = f(x_k^{l_k}) \leq f(x_k^{l_k-1}) - c\alpha_k^2 \leq f(x_{k-1}^0) - l_k c\alpha_k^2 = f(x_{k-1}) - l_k c\alpha_k^2.$$

This immediately implies that the sequence $\{f(x_k)\}_{k\geq 0}$ is non-increasing, and also gives the bound (6). Finally, (7) follows by substituting the second estimate in (6) into (5) and using the fact that $\alpha_k = \alpha_0/2^k$ for $k \geq 1$. □

Later on, we shall also use the bound (7) in the complexity analysis of Algorithm 1 applied to the minimization of a smooth (and possibly nonconvex) function f. In the convex and strongly convex cases, a much better bound can be obtained since, as we shall argue, one can establish tighter bounds on l_k than (6).

2.2 Connection with directional direct search

Our method is very similar to a "standard" directional direct search method; see for instance [19]. Let us outline some of the similarities and differences:

- The method in [19] contains an additional "search step" as an option for the user. This step is not specified beyond the requirement that one "somehow" identifies a point x (not necessarily obtained by moving along direction in D) leading to sufficient decrease. This step is traditionally included – but is skipped in the complexity analysis for obvious reasons (no specification is given for how it should be performed). For the sake of exposition and clarity, and since this step does not influence the complexity analysis, our method does not include this step.

- Our update rule for the stepsize is a bit different from that in [19]. We halve the stepsize before each iteration (it is possible to replace the factor of two by any factor $\gamma > 1$; the analysis extends in a straightforward way), and keep it constant within each iteration until a new iterate is computed. Also, we never increase the stepsize parameter. On the other hand, the method of Vicente changes the stepsize within each iteration: everytime a new point x_k^l is computed. Moreover, the rules for the change are more flexible: the stepsize is always decreased, by a factor $t \in [\beta_1, \beta_2]$ (where $0 < \beta_1 \leq \beta_2 < 1$) after an unsuccessful step, and is kept unchanged or increased by a factor $t \in [1, \gamma]$ after each successful step. These constants then appear in the complexity analysis. We utilize a simpler update rule, as that is all we need for the complexity analysis.

- The method in [19] uses an arbitrary nonnegative "forcing function" $\rho(\alpha)$ in place where we use $c\alpha^2$. However, neither the analysis in[19], nor our analysis, benefit from the usage of a different forcing function: and the best results are obtained for $\rho(\alpha) = c\alpha^2$.
- In addition to the above differences, our analysis is substantially different, vastly briefer and leads to more compact, interpretable and sharper complexity results.

3 Main Results

In this section we state three complexity results, covering the nonconvex, convex and strongly convex case, and provide a brief discussion of the results. The proofs of the complexity theorems can be found in the next section.

3.1 Assumptions

In this section we describe the assumptions that are made throughout the paper.

Assumption 2 (Boundedness of f)**.** $f^* \overset{def}{=} \inf\{f(x) \;\; x \in \mathbb{R}^n\} > -\infty$.

Recall that we already needed this assumption in order to establish Proposition 1.

Assumption 3 (Smoothness of f)**.** *f is L-smooth. That is, f has a Lipschitz continuous gradient, with a positive Lipschitz constant L:*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \textit{for all} \quad x, y \in \mathbb{R}^n. \tag{8}$$

Section 3.2 deals with the most general case – we need not impose any additional assumptions beyond L-smoothness and boundedness. In Section 3.3 we further assume that f is convex and has bounded level sets. In Section 3.4 we further assume that f is strongly convex.

Assumption 4 (Properties of D)**.** *D is a finite set of unit-length vectors and*

$$\mu \overset{def}{=} \min_{0 \neq v \in \mathbb{R}^n} \max_{d \in D} \frac{\langle v, d \rangle}{\|v\|} > 0,$$

where $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product and $\|\cdot\|$ is the standard Euclidean norm.

The assumption that the *cosine measure*, μ, is positive, is equivalent (via a simple separation argument) to the requirement that non-negative linear combinations of vectors from D span $\mathbb{R}^n$. Sets with this property are called *positive spanning sets*. This assumption is standard in the literature on direct search. Indeed, it is clearly necessary as otherwise it is not possible to guarantee that any point (and, in particular, the optimal point) can be reached by a sequence of steps of the algorithm.

Note that we assume that all vectors in D are of unit length. While the algorithm and theory can be extended in a straightforward way to allow for vectors of different lengths (which, in fact, is standard in the literature), this does not lead to an improvement in the complexity bound and

merely makes the analysis and results a bit less transparent. Hence, the unit length assumption is enforced for convenience.

The cosine measure μ has a straightforward geometric interpretation: for each nonzero vector v, let $d \in D$ be the vector forming the smallest angle with v and let $\mu(v)$ be the cosine of this angle. Then $\mu = \min_v \mu(v)$. That is, for every nonzero vector v there exists $d \in D$ such that the cosine of the angle between these two vectors is at least $\mu > 0$ (i.e., the angle is acute). In the analysis, we shall consider the vector v to be the negative gradient of f at the current point. While this gradient is unknown, we know that there is a direction in D which approximates it well, with the size of μ being a measure of the quality of that approximation: the larger μ is, the better.

Equivalently, μ can be seen as the largest scalar such that for all nonzero v there exists $d \in D$ so that the following inequality holds:

$$\mu \|v\| \|d\| \leq \langle v, d \rangle . \tag{9}$$

This is a reverse of the Cauchy-Schwarz inequality, and hence, necessarily, $\mu \leq 1$. However, for $\mu = 1$ to hold we would need D to be dense on the unit sphere. For better insight, consider the following example. If D is chosen to be the "maximal positive basis" (composed of the coordinate vectors together with their negatives: $D = \{\pm e_i \mid i = 1, \ldots, n\}$), then

$$\mu = \frac{1}{\sqrt{n}}. \tag{10}$$

3.2 Nonconvex case

In this section, we state our most general complexity result – one that does not require any additional assumptions on f, besides smoothness and boundedness. In particular, it applies to nonconvex objective functions.

Theorem 5 (Nonconvex case)**.** *Let Assumptions 2, 3 and 4 be satisfied. Choose initial iterate $x_0 \in \mathbb{R}^n$, initial stepsize parameter $\alpha_0 > 0$, error tolerance $\epsilon > 0$, and iteration counter*

$$k(\epsilon) = \left\lceil \log_2 \left(\frac{\alpha_0 (L/2 + c)}{\mu \epsilon} \right) \right\rceil . \tag{11}$$

Then, $\left\| \nabla f(x_{k(\epsilon)}) \right\| \leq \epsilon$. Moreover, the algorithm performs in total at most

$$N(k(\epsilon)) \leq 1 + |D| \left(k(\epsilon) + \frac{16 (f(x_0) - f^*)(\frac{L}{2} + c)^2}{3 c \mu^2 \epsilon^2} \right) \tag{12}$$

function evaluations.

We shall now briefly comment the above result.

- In the algorithm we have freedom in choosing c. It is easy to see that the choice $c = \frac{L}{2}$ minimizes the dominant term in the complexity bound (12), in which case the bound takes the form

$$\mathcal{O} \left(\frac{L |D| (f(x_0) - f^*)}{\mu^2 \epsilon^2} \right) . \tag{13}$$

  Needless to say, in a derivative-free setting the value of L is usually not available.

- If D is chosen to be the "maximal positive basis" (see (10)), the bound (13) reduces to

$$\mathcal{O}\left(\frac{n^2 L(f(x_0)-f^*)}{\epsilon^2}\right).$$

3.3 Convex case

In this section, we analyze the method under the additional assumption that f is convex. For technical reasons, we also assume that the problem is solvable (i.e., that it has a minimizer x_*) and that, given an initial iterate $x_0 \in \mathbb{R}^n$, the quantity

$$R_0 \stackrel{\text{def}}{=} \sup_{x\in\mathbb{R}^n} \{\|x - x_*\| \ : \ f(x) \le f(x_0)\} \tag{14}$$

is finite. Further, define

$$B \stackrel{\text{def}}{=} \frac{R_0(c+\frac{L}{2})}{\mu}. \tag{15}$$

We are now ready to state the complexity result.

Theorem 6 (Convex case)**.** *Let Assumptions 2, 3 and 4 be satisfied. Further assume that f is convex, has a minimizer x_* and $R_0 < \infty$ for some initial iterate $x_0 \in \mathbb{R}^n$. Assume that the initial stepsize is large enough: $\alpha_0 \ge (f(x_0)-f(x_*))/B$. Then iterates of Algorithm 1 satisfy*

$$f(x_k) - f(x_*) \le \frac{B\alpha_0}{2^k}, \quad \|\nabla f(x_k)\| \le \frac{(c+\frac{L}{2})\alpha_0}{2^k\mu}, \quad k \ge 0,$$

where at iteration k the method needs to perform at most $|D|\left(1+\frac{2^{k+1}B}{c\alpha_0}\right)$ function evaluations.

In particular, if we set $k = k(\epsilon) \stackrel{def}{=} \left\lceil \log_2\left(\frac{B\alpha_0}{\epsilon}\right)\right\rceil$, then $f(x_k) - f(x_*) \le \epsilon$, while the method performs in total at most

$$N(k(\epsilon)) \le 1 + |D|\left(k(\epsilon) + \frac{8B^2}{c\epsilon}\right) \tag{16}$$

function evaluations.

We shall now comment the result.

- Again, we have freedom in choosing c (and note that c appears also in the definition of B). It is easy to see that the choice $c = \frac{L}{2}$ minimizes the dominating term $\frac{B^2}{c}$ in the complexity bound (16), in which case $B = \frac{LR_0}{\mu}$, $\frac{B^2}{c} = \frac{2LR_0^2}{\mu^2}$ and the bound (16) takes the form

$$1 + |D|\left[\left\lceil \log_2\left(\frac{LR_0\alpha_0}{\mu\epsilon}\right)\right\rceil + \frac{16LR_0^2}{\mu^2\epsilon}\right] = \mathcal{O}\left(\frac{LR_0^2|D|}{\mu^2\epsilon}\right). \tag{17}$$

- If D is chosen to be the "maximal positive basis" (see (10)), the bound (17) reduces to

$$\mathcal{O}\left(\frac{n^2 LR_0^2}{\epsilon}\right).$$

- It is possible to improve the algorithm by introducing an additional stopping criterion: $l_k \ge \frac{B}{c\alpha_k}$. The analysis is almost the same, and resulting number of function evaluations is halved in this case. However, this improvement is rather theoretical, since we typically do not know the value of B.

3.4 Strongly convex case

In this section we introduce an additional assumption: f is λ-strongly convex for some (strong convexity) constant $\lambda > 0$. That is, we require that $\forall x, y \in \mathbb{R}^n$, we have

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\lambda}{2}\|y - x\|^2. \tag{18}$$

In particular, by minimizing both sides of the above inequality in y, we obtain the standard inequality

$$f(x) - f(x_*) \leq \frac{1}{2\lambda}\|\nabla f(x)\|^2. \tag{19}$$

In what follows, we will make use of the following quantity:

$$S \stackrel{\text{def}}{=} \frac{(c + \frac{L}{2})^2}{2\lambda\mu^2}. \tag{20}$$

Theorem 7 (Strongly convex case)**.** *Let Assumptions 3 and 4 be satisfied. Assume that f is λ-strongly convex. Further assume that the initial stepsize is large enough: $\alpha_0 \geq \sqrt{f(x_0) - f(x_*)/S}$. Then the iterates of Algorithm 1 satisfy*

$$f(x_k) - f(x_*) \leq S\left(\frac{\alpha_0}{2^k}\right)^2, \quad \|\nabla f(x_k)\| \leq \frac{(c + \frac{L}{2})\alpha_0}{2^k\mu}, \quad k \geq 0,$$

where at iteration $k \geq 1$ the method needs to perform at most $|D|\left(\frac{4S}{c} + 1\right)$ function evaluations.

In particular, if we set

$$k = k(\epsilon) \stackrel{def}{=} \left\lceil \log_2\left(\alpha_0\sqrt{\frac{S}{\epsilon}}\right)\right\rceil,$$

then $f(x_k) - f(x_*) \leq \epsilon$, while the method performs in total at most

$$N(k(\epsilon)) \leq 1 + |D|\left(\frac{4S}{c} + 1\right)\left(1 + \log_2\left(\alpha_0\sqrt{\frac{S}{\epsilon}}\right)\right) \tag{21}$$

function evaluations.

Let us now comment on the result.

- As before, in the algorithm we have freedom in choosing c. Choosing $c = \frac{L}{2}$ minimizes the dominating term $\frac{S}{c}$ in the complexity bound (16), in which case $S = \frac{L^2}{2\lambda\mu^2}$, $\frac{S}{c} = \frac{L}{\lambda\mu^2}$ and the bound (21) takes the form

$$1 + |D|\left(1 + \frac{4L}{\lambda\mu^2}\right)\left(1 + \log_2\left(\frac{\alpha_0 L}{\mu}\sqrt{\frac{1}{2\lambda\epsilon}}\right)\right). \tag{22}$$

- If D is chosen to be the "maximal positive basis" (see (10)), the bound (22) reduces to

$$\mathcal{O}\left(\frac{n^2 L}{\lambda}\log_2\left(\frac{nL^2\alpha_0^2}{\lambda\epsilon}\right)\right) = \tilde{\mathcal{O}}\left(n^2\frac{L}{\lambda}\right),$$

  where the $\tilde{\mathcal{O}}$ notation suppresses the logarithmic term. The complexity is proportional to the condition number L/μ.

- As in the convex case, we can introduce the additional stopping criterion $l_k \leq \frac{3S}{c}$. The analysis is similar and the bound on function evaluation can be reduced by the factor of 4/3. However, in practice we often do not know S.

3.5 Further Discussion

Besides the starting point x_0, our algorithm has only two parameters: initial stepsize parameter $\alpha_0 > 0$ and forcing constant $c > 0$.

While, as we have seen, it is theoretically optimal to choose $c = L$, this need not be done[1], since our complexity results hold for any $c > 0$.

However, note that our complexity results in the convex and strongly convex cases hold under the assumption that α_0 is sufficiently large, in relation to some unknown instance-specific quantities. So, it may seem that our method needs the knowledge of these parameters to properly set the value of α_0. We shall see in Section 5 that it is possible to initialize, at small cost which is dominated by the cost of the algorithm, the value of α_0, so that the theoretical requirement is satisfied. With this initialization, the same algorithm, with identical parameter settings, is suitable for nonconvex, convex and strongly convex functions. Thanks to this property, the algorithm is adaptive to local properties of f without any parameter tuning.

4 Proofs

We now prove the theorems stated in the previous section.

We start with a technical result (Lemma 1) used repeatedly in our analysis. While this result is standard in the analysis of direct search methods[2], we shall use it in a novel way, which leads to a vast simplification of the analysis (2 pages in total for all three proofs) and to sharper and cleaner complexity bounds.

Lemma 1 ([7]). *Fix $x \in \mathbb{R}^n$ and $\alpha > 0$. If $f(x) - c\alpha^2 < f(x + \alpha d)$ for all $d \in D$, then*

$$\|\nabla f(x)\| \leq \frac{1}{\mu}\left(\frac{L}{2} + c\right)\alpha. \tag{23}$$

Proof. Since f is L-smooth, (8) implies that for all $x, y \in \mathbb{R}^n$ we have $f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2$. It then follows that for all $d \in D$, we have $-f(x + \alpha d) < -f(x) + c\alpha^2$. Summing these two inequalities, and setting $y = x + \alpha d$, we obtain

$$0 < \langle \nabla f(x), \alpha d\rangle + c\alpha^2 + \frac{L}{2}\|\alpha d\|^2. \tag{24}$$

[1] In fact, in the derivative-free setting this can't be done, since usually L will not be known.

[2] Lemma 1 is usually stated in the general setting with the vectors in D allowed to be of arbitrary lengths, and with $c\alpha^2$ replaced by an arbitrary *forcing function* $\rho(\alpha)$. In this paper we choose to present the result with $\rho(\alpha) = c\alpha^2$ since i) the complexity guarantees do not improve by considering a different forcing function, and because ii) the results and proofs become a bit less transparent. For a general forcing function, the statement would say that if $f(x) - \rho(\alpha) < f(x + \alpha d)$ for all $d \in D$, then

$$\|\nabla f(x)\| \leq \mu^{-1}\left(\tfrac{L}{2}\alpha d_{max} + \tfrac{\rho(\alpha)}{\alpha}\tfrac{1}{d_{min}}\right),$$

where $d_{min} = \min\{\|d\| \,:\, d \in D\}$ and $d_{max} = \max\{\|d\| \,:\, d \in D\}$. In this form, the lemma is presented, for instance, in [7].

Let $d \in D$ be such that $\mu\|\nabla f(x)\|\|d\| = \mu\|\nabla f(x)\| \leq -\langle \nabla f(x), d\rangle$ (see (9)). It only remains to multiply this inequality by α, add it to (24) and rearrange the result. □

Now we are ready to prove the three complexity theorems.

4.1 Proof of Theorem 5 (Nonconvex case)

We first argue that if $\alpha_k \leq \frac{\mu\epsilon}{L/2+c}$ for some $k \geq 1$, then $\|\nabla f(x_k)\| \leq \epsilon$. We first note that since f is bounded below, from (6) we know that l_k is bounded from above for each k and hence the method indeed produces an infinite sequence of iterates $\{x_k\}_{k\geq 0}$. Indeed, since by construction of x_k we have $f(x_k+\alpha_k d) > f(x_k) - c\alpha_k^2$ for all $d \in D$, Lemma 1 implies that $\|\nabla f(x_k)\| \leq \frac{1}{\mu}(L/2+c)\alpha_k \leq \epsilon$. On the other hand,

$$\alpha_{k(\epsilon)} = \frac{\alpha_0}{2^{k(\epsilon)}} \overset{(11)}{\leq} \frac{\mu\epsilon}{L/2+c}.$$

The bound (12) is obtained by using inequality (7) with $k = k(\epsilon)$.

4.2 Proof of Theorem 6 (Convex case)

The proof is based on the following simple observation, formulated as a lemma.

Lemma 2. *For all $x \in \mathbb{R}^n$ such that $f(x) \leq f(x_0)$ and for all $\alpha > 0$, one of the following holds:*

1. $f(x+\alpha d) \leq f(x) - c\alpha^2, \quad$ *for some $d \in D$,*
2. $f(x) - f(x_*) \leq B\alpha$.

Proof. If 1) does not hold, then by convexity of f, Cauchy-Schwarz inequality and Lemma 1, we have $f(x) - f(x_*) \leq \langle \nabla f(x), x - x_*\rangle \leq \|\nabla f(x)\|\|x - x_*\| \leq B\alpha$, where the last inequality follows from (23), (14) and (15). □

We now proceed to the proof of Theorem 6. Let $r(x) \overset{\text{def}}{=} f(x) - f(x_*)$. We first claim that for all k, $r(x_k) \leq B\alpha_k$. This clearly holds because by construction of x_k we know that $f(x_k + \alpha_k d) > f(x_k) - c\alpha_k^2$ for all $d \in D$, and hence Lemma 2 implies that $r(x_k) = r(x_k^{l_k}) \leq B\alpha_k$. Likewise, Lemma 1 (we can apply the lemma since by Proposition 1 $f(x_k) \leq f(x_0)$) implies that

$$\|\nabla f(x_k)\| \leq \frac{(c + L/2)\alpha_k}{\mu} = \frac{(c+L/2)\alpha_0}{2^k\mu}.$$

Now, let give a bound on l_k. Note that $0 \leq r(x_k) \leq r(x_{k-1}) - l_k c\alpha_k^2 \leq 2B\alpha_k - l_k c\alpha_k^2$, whence we have the bound

$$l_k \leq \frac{2B}{c\alpha_k} = \frac{2^{k+1}B}{c\alpha_0}. \tag{25}$$

We can now estimate the total number of function evaluations as follows:

$$\begin{aligned} N(k(\epsilon)) &\overset{(5)}{\leq} 1+\sum_{k=1}^{k(\epsilon)}|D|(l_k+1) \\ &\overset{(25)}{\leq} 1+|D|\sum_{k=1}^{k(\epsilon)}\left(\frac{2B}{c\alpha_k}+1\right) \\ &= 1+|D|k(\epsilon)+\frac{2B|D|}{c\alpha_0}\sum_{k=1}^{k(\epsilon)}2^k \\ &\leq 1+|D|k(\epsilon)+\frac{2B|D|}{c\alpha_0}2^{k(\epsilon)+1} \quad \leq \quad 1+|D|\left(k(\epsilon)+\frac{8B^2}{c\epsilon}\right). \end{aligned}$$

4.3 Proof of Theorem 7 (Strongy convex case)

The proof is based on a similar auxiliary result to the one we have employed in the previous section.

Lemma 3. *For all $x\in\mathbb{R}^n$ and $\alpha>0$, one of the following holds:*

1. $f(x+\alpha d)\leq f(x)-c\alpha^2, \quad$ *for some $d\in D$,*
2. $f(x)-f(x_*)\leq S\alpha^2$.

Proof. If 1) does not hold, then by strong convexity of f and Lemma 1, we have

$$f(x)-f(x_*)\overset{(19)}{\leq}\tfrac{1}{2\lambda}\|\nabla f(x)\|^2\overset{(23)+(20)}{\leq}S\alpha^2,$$

which finishes the proof. □

We now proceed to the proof of Theorem 7. Let $r(x)\overset{\text{def}}{=}f(x)-f(x_*)$. We first claim that for all k, $r(x_k)\leq S\alpha_k^2$. This clearly holds because by construction of x_k, we know that $f(x_k+\alpha_k d)>f(x_k)-c\alpha_k^2$ for all $d\in D$, and hence Lemma 3 implies that $r(x_k)=r(x_k^{l_k})\leq S\alpha_k^2$. Moreover, Lemma 1 directly implies $\|\nabla f(x_k)\|\leq\frac{1}{\mu}(L/2+c)\alpha_k=\frac{1}{\mu}(L/2+c)\alpha_0/2^k$.

Now, let us derive a bound on l_k. Since $r(x_{k-1})\leq S\alpha_{k-1}^2=4S\alpha_k^2$, and in each step we decrease the function value by at least $c\alpha_k^2$, we must have $l_k\leq l:=\frac{4S}{c}$. In view of (5), the total number of function evaluations can be bounded above by $N(k(\epsilon))\leq 1+k(\epsilon)|D|(l+1)$, leading to (21).

5 Initialization

In Theorems 6 and 7 we impose a condition on α_0, requiring that α_0 be large enough. Note however that there is no straightforward way of checking whether the condition is satisfied as the quantities involved $(R_0,L,\lambda,f(x_*))$ are not known.

5.1 Efficient strategy: keep x_0 fixed, search for α_0

In this section we propose an efficient algorithm for finding suitable α_0.

Observe that if we find $\alpha_0 > 0$ such that

$$f(x_0 + \alpha_0 d) > f(x_0) - c\alpha_0^2 \quad \text{for all} \quad d \in D, \tag{26}$$

then Lemmas 2 and 3 imply that α_0 satisfies the assumption Theorems 6 and 7, respectively. So, we shall devise a method for finding such α_0 – this is what Algorithm 2 does. We shall prove that the algorithm does what is is supposed to do, and we also give a complexity bound from which it will be clear that the method is very efficient.

At the same time, we would like to be able to use this initialization algorithm also in the nonconvex case – simply because in derivative-free optimization we may not know whether the function we are minimizing is or is not convex! Note that in the nonconvex case our complexity theorem does not require any assumption on α_0 to be satisfied. So, in this case, we would like to simply be able to say that the initialization algorithm stops after a certain (small) number of function evaluations, and be able to say something about α_0.

Algorithm 2 Initialization Method for Finding α_0

1. INPUT: $x_0 \in \mathbb{R}^n$; initial estimate $\bar{\alpha} > 0$ of α_0; forcing constant $c > 0$; $D = \{d_1, d_2, \ldots, d_p\}$
2. Initialize $i \leftarrow 1$ and $\alpha \leftarrow \bar{\alpha}$
3. **while** $i \leq |D|$
   - **if** $f(x_0 + \alpha d_i) \leq f(x_0) - c\alpha^2$ **then** set $\alpha \leftarrow 2\alpha$
   - **else** $i \leftarrow i + 1$
4. OUTPUT: $\alpha_0 = \alpha$

The following theorem describes the behaviour of the Initialization Method.

Theorem 8. *Let Assumption 2 be satisfied (i.e., f is bounded below by $f^* > -\infty$). Then Algorithm 2 outputs α_0 satisfying*

$$\bar{\alpha} \leq \alpha_0 \leq \max\left\{\bar{\alpha}, 2\sqrt{\frac{f(x_0) - f_*}{c}}\right\} \stackrel{def}{=} M,$$

and performs in total at most

$$|D| + \log_2(M/\bar{\alpha}) \tag{27}$$

function evaluations (not counting the evaluation of $f(x_0)$).

Let, moreover, the Assumptions of Theorem 6 (resp. Theorem 7) hold (but, of course, we do not require the condition on α_0 to hold). Then α_0—the output of Algorithm 2—satisfies condition (26)*, and hence it also satisfies the condition prescribed in Theorem 6 (resp. Theorem 7).*

Proof. If at some point during the execution of the algorithm we have $\alpha > \sqrt{(f(x_0) - f_*)/c} \stackrel{\text{def}}{=} h$, then the "if" condition cannot be satisfied, and hence α will not be further doubled. So, if $\bar{\alpha} \leq h$,

then $\alpha \leq 2h$ for all α generated throughout the algorithm, and if $\bar{\alpha} > h$, then $\alpha = \bar{\alpha}$ throughout the algorithm. Consequently, $\alpha \leq \max\{\bar{\alpha}, 2h\}$ throughout the algorithm.

Now note that at each step of the method, either α is doubled, or i is increased by one. Since, as we have just shown, α remains bounded, and D is finite, the algorithm will stop. Moreover, the method performs function evaluation of the form $f(x_0 + \alpha d_i)$, where α can assume at most $1 + \log_2(M/\bar{\alpha})$ different values and d_i at most $|D|$ different values, in a fixed order. Hence, the method performs at most $|D| + \log_2(M/\bar{\alpha})$ function evaluations (not counting $f(x_0)$). This proves the first part of the theorem.

Let us now establish the second part of the theorem. In particular, f is convex. Note for each $d_i \in D$ there exists $\alpha_i \leq \alpha_0$ for which

$$f(x_0 + \alpha_i d_i) > f(x_0) - c\alpha_i^2. \tag{28}$$

Indeed, this holds for α_i equal to the value of α at the moment when the index i is increased. We now claim that, necessarily, inequality (28) must hold with α_i replaced by α_0. We shall show that this follows from convexity. Indeed, by convexity,

$$\frac{f(x_0 + \alpha_0 d_i) - f(x_0)}{\alpha_0} \geq \frac{f(x_0 + \alpha_i d_i) - f(x_0)}{\alpha_i},$$

which implies that

$$f(x_0 + \alpha_0 d_i) \geq f(x_0) + (f(x_0 + \alpha_i d_i) - f(x_0)) \frac{\alpha_0}{\alpha_i} > f(x_0) - c\alpha_i^2 \frac{\alpha_0}{\alpha_i} \geq f(x_0) - c\alpha_0^2.$$

We have now established (26), and hence the second statement of theorem is proved.

□

The runtime of Algorithm 2, given by (27), is negligible compared to the runtime of Algorithm 1. Hence, initialization of stepsize is not a practical issue.

5.2 A bootstrapping initialization strategy: keep α_0 fixed, search for x_0

An alternative way of having the bound on α_0 satisfied is to keep α_0 unchanged and instead search for a suitable starting point x_0. Consider running Algorithm 1, from x_0 (considered to be unsuitable), using stepsize α_0, for one iteration, producing point x_1. Note that, by construction, the condition (26) is satisfied if we replace x_0 by x_1. The cost of this initialization strategy is equal to the cost of running Algorithm 1 for 1 iteration, which is $|D|(l_1 + 1)$, where from (6) we have $l_1 \leq (f(x_0) - f_*)/(c\alpha_0^2)$.

At first sight, this cost seems negligible: after all, it is just a single iteration of the method. However, since α_0 and x_0 were not good in the first place, the cost of even this single iteration can be excessive. Indeed, l_1 can potentially be a large quantity – this depends on how our choice of c and α_0 compares with the residual $f(x_0) - f_*$. If $c\alpha^2$ is too small, then l_1 can be too large–much larger than the overall complexity of the method with initialization via Algorithm 2.

References

[1] M A Abramson and C Audet. Convergence of mesh adaptive direct search to second-order stationary points. *SIAM Journal on Optimization*, 17:606–619, 2006.

[2] C Audet and J E Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006.

[3] C Audet and D Orban. Finding optimal algorithmic parameters using derivative-free optimization. *SIAM Journal on Optimization*, 17:642–664, 2006.

[4] J E Audet, Cand Dennis Jr. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2002.

[5] C Cartis, N I M Gould, and P L Toint. On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 20(6):2833–2852, 2010.

[6] C Cartis, N I M Gould, and P L Toint. On the oracle complexity of first-order and derivative-free algorithms for smooth nonconvex minimization. *SIAM Journal on Optimization*, 22(1):66–86, 2012.

[7] A R Conn, K Scheinberg, and L N Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization. SIAM, 2009.

[8] M Dodangeh and L N Vicente. Worst case complexity of direct search under convexity. Technical report, Technical Report 13-10, Dept. Mathematics, Univ. Coimbra, 2013.

[9] E D Dolan, R M Lewis, and V Torczon. On the local convergence of pattern search. *SIAM Journal on Optimization*, 14(2):567–583, 2003.

[10] J C Duchi, M I Jordan, M J Wainwright, and A Wibisono. Optimal rates for zero-order optimization: the power of two function evaluations. *arXiv:1312.2139*, 2013.

[11] R Garmanjani and L N Vicente. Smoothing and worst case complexity for direct-search methods in non-smooth optimization. *IMA Journal of Numerical Analysis*, 2012.

[12] R Hooke and T A Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229, 1961.

[13] T G Kolda, R M Lewis, and V Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45:385–482, 2003.

[14] Yu Nesterov. *Introductory lectures on convex optimization: A basic course*. Kluwer Academic Publishers, 2004.

[15] Yu Nesterov. Random gradient-free minimization of convex functions. Technical report, Université catholique de Louvain, CORE Discussion Paper 2011/16, 2011.

[16] B Recht, K G Jamieson, and R Nowak. Query complexity of derivative-free optimization. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2012.

[17] P Richtárik and M Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

[18] V Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7:1–25, 1997.

[19] L N Vicente. Worst case complexity of direct search. *EURO Journal on Computational Optimization*, 1(1-2):143–153, 2013.

[20] Yu Wen-ci. Positive basis and a class of direct search techniques. *Scientia Sinica, Special Issue of Mathematics*, 1:53–67, 1979.